TLN - Mazzei

Mario Bifulco

a.a. 2022-2023

Indice

Parte I

introduzione

Punti salienti della storia del NLP:

- 1953: IBM mostra un primo sistema di traduzione
- 1966: ALPAC, NLP winter
- 2007: Google translate statistical method
- 2011: Hey Siri (basato su ontologie)
- 2017: Google translate NN method
- 2017: Microsoft crea un programma per speech recognition che raggiunge l'uomo

Capitolo 1

Linguaggio umano

Caratteristiche:

- Discretezza, il linguaggio umano si basa su atomi discreti, come fonemi, morfemi e parole
- Ricorsività, le frasi si possono concatenare in modo dipendente aumentado la complessità
- Dipendenza dalla struttura, parole nello stesso "ordine" possono avere significati diversi in base alla struttura grammaticale
- Località, la posizione degli elementi nella frase è rilevante

Turing propone il test per verificare l'intelligenza delle macchine, questo test è fortemente basato sul linguaggio

Problemi:

- Il test varia molto in base al giudice
- Problema della stanza cinese

Nel 1966 viene proposto un sistema a regole Eliza che interpreta uno psicologo, Emacs

Il test di Turing si evolve nel test **Loebner**, il sistema esperto deve per 5 minuti parlare e fare battute. In questo caso i giudici erano consapevole di parlare con un sistema, **Problema della Pareidolia**, l'uomo cerca l'uomo anche quando non c'è

Un'altra evoluzione è il test di **Winograd**, basato sul ragionamento comune, facile da verificare e facile da svolgere per gli uomini, non sono richiesti giudici. Nel 2016 i sisitemi superavano al 58%, nel 2019 i sistemi BERT hanno ottenuto 90.01%

Il linguaggio umana è caratterizzato da AMBIGUITÀ, essa permette una comunicazione molto efficiente ma aumenta di molto la complessità per i sistemi automatici

1.1 Proprietà notevoli

Il linguaggio non è standard ed evolve sempre con neologismo

Inoltre possiamo incontrare fenomeni di segmentazione (un significato diviso in più parole della frase) e di locuzioni (parole che hanno senso nel contesto solo se prese insieme)

Inoltre noi basiamo la nostra comunicazione sulla conoscenza del mondo

1.2 ChatGPT

Sistema di dialogo basato su tre fasi di apprendimento, unsupervised, supervised e reinforcement

Può tornare utile in molti contesti, ma bisogna sapere i limiti

Secondo *Noam Chomsky* è un plagio High-Tech, un sistema che evita l'apprendimento

Capitolo 2

Sistemi disponibili

2.1 Sistemi a regole vs Sistemi statistici

Inizialmente la battaglia era tra sistemi a regole e sistemi statistici basati su dati

Ad esempio il '.' dava problemi di riconoscimento (numeri, abbreviazioni, e fine frase)

Text segmentation, aggiunge un tag di **EoS** alla fine della frase

I sistemi statistici basati su machine learning costruiscono un albero (ogni nodo corrisponde ad una feature) e in base a quello decidono come classificare la frase

Ma per caratteri speciali (come '.') le feature possono essere molto complesse

La scelta delle feature diventa cruciale

Attualmente i sistemi migliori di sentence splitting fanno 1.6% d'errore

2.2 Sistemi neurali

Con l'uso dei sistemi neurali non sono scritte a mano le singole feature, ma sono apprese dal sistema, la ricerca è focalizzata sull'architettura migliore

Parte II

Livelli linguistici

L'ipotesi di lavoro fondamentale è che supponiamo che i livelli linguistici si possano processare a cascata senza perdita di informazioni

Sappiamo non essere così per l'uomo, ma la complessità derivata dal trattare tutto insieme è troppo elevata (i primi esperimenti erano in questa direzione)

L'obiettivo è convertire una frase in modo che sia comprensibile per la macchina e si possa fare ragionamento automatico

Capitolo 3

Fonetica

Il livello della fonetica/fonologia prevede di capire il suono di una voce e riportarlo a testo, analogamente dato un testo in input si richiede di rirpodurre i suoni corrispondenti

Questo tipo di task è ad oggi considerato risolto e ricade in un problema quasi esclusivamente ingegneristico

Capitolo 4

Morfologia

Compresione delle singole parole e delle parole composte

Parola: sequenza di caratteri delimitata da spazi o punteggiatura

Problemi: ci sono parole composte da diverse sotto-particelle, non sono realmente parole nuove, ma il loro utilizzo è molto comune nelle lingue

Ipotesi della semantica lessicale è che le parole in un contesto dispongano di un significato unitario, tuttavia ci sono molti esempi di pezzi di frase che sarebbe meglio considerare come un unico blocco

Inoltre ci sono diverse forme di ambiguità introdotte dai suffissi (in base al suffisso la parola potrebbe assumere significati diversi, ma la radice di significato resta la stessa)

La struttura dati di riferimento è la **lista**

4.1 Classi aperte e chiuse

Le classi chiuse sono elementi della grammatica che "non cambiano" nel tempo, o comunque cambiano in archi temporali dilatati

Le classi aperte sono parti della lingua che cambiano in tempi brevi, solitamente vengono estese con nuovi termini e quelli vecchi col tempo possono diventare desueti

4.2 PoS e NER tagging

4.2.1 Part of Speech

Dalla "grammatica" si possono estrarre parti del discorso per disambiguare (almeno in parte) parole composte

I PoS permettono di ricostruire rapporti sintagmatici, ovvero relazioni tra parole di una stessa frase

Molte parole hanno una sola PoS (85%), quindi si disambiguano automaticamente, il problema è che il 60% delle parole comuni sono quelle 'ambigue', ovvero con diversi significati e PoS associati

Possono essere usati per il text-to-speech (linea guida per la pronuncia di termini ambigui) o per la traduzione (ordinamento corretto delle parole), inoltre se si scrivono regex basate su PoS si ottengono regole più robuste e versatili

Ad ogni parola si associa un tag, il sistema ha in input una lista di parole e restituisce la lista di parole e la lista di tag associati

Lo stato dell'arte per il PoS tagging è del 97%, il problema è che la baseline ottiene il 92% (tagga tutto col tag più frequente e metti le parole ignote come NOUN)

I set di PoS possono essere più o meno estesi in base alla granularità che si vuole raggiungere

Ad esempio il Penn TreeBank è molto fine e quindi più informativo, mentre il set Universal Dependencies di Google è meno preciso ma più versatile

Algoritmi per il PoS tagging

- Hidden Markov Model
- Maximum Entropy Markov Models
- Neural sequence models (RNN e transformer)
- LLM (BERT), finetuned

Tutti questi metodi richiedono training set annotati a mano e performano in modo analogo

4.2.2 Tag parole ignote

Le parole non conosciute possono aumentare gli errori del 1-2%

Metodi di mitigazione:

1. assumere siano NOUN
2. assumere una distribuzione uniforme dei PoS
3. usare un dizionare esterno come morph-it
4. usare informazioni di tipo morfologico
5. trattarle come sconosciute (tag a parte con la sua probabilità e distribuzione simile al resto del training set)

4.2.3 Sistemi a regole

Idea, assegna tutti i possibili tag alle parole e si procede per eliminazione tramite regole

L'ordine di grandezza è di circa 1000 regole (possono essere prodotte tramite machine learning)

4.2.4 Sistemi stocastici

Si cerca la sequenza di tag più probabile data l'osservazione (sequenza di parole)

Modelling: produrre un modello

Learning: algoritmo per apprendere i parametri

Deconding: algoritmo per applicare il modello e computare i risultati

HMM

Vogliamo $\hat{t}_1^n = argmax_{t_1^n} P(t_1^n|w_1^n)$

Modelling Usiamo la regola bayesiana per trasformate l'equazione in una versione più semplice

$$\hat{t}_1^n = argmax_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)$$

Dove $P(w_1^n|t_1^n) \approx \prod_{i=1}^n P(w_i|t_i)$ e $P(t_1^n) \approx \prod_{i=1}^n P(t_i|t_{i-1})$, quindi:

$$\hat{t}_1^n = argmax_{t_1^n} P(w_1^n|t_1^n)P(t_1^n) \approx argmax_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Questo sistema apprende due tipi di probabilità, la prima che predice il tag data la parola in input, la seconda che stima il tag prendendo in considerazione il tag precedente

Deconding Per usare il modello possiamo enumerare i possibili percorsi e restituire quello con probabilità maggiore

Problema, ci sono tanti possibili percorsi possibili $nums\_tags^{nums\_words}$

Una possibile soluzione è usare la programmazione dinamica Algoritmo di Viterbi

Crea una matrice dove le colonne corrispondono agli input (possibili parole) e le righe sono i tag

$$v_t(j) = max_{i=1}^N v_{t-1}(i)a_{ij}b_j(o_t)$$

Dove $v_{t-1}(i)$ le probabilità dei path di Viterbi al tempo precedente, a_{ij} è la probabilità di transizione da q_i a q_j, $b_j(o_t)$ è la probabilità di osserare il simbolo o_t dato lo stato corrente j

Algorithm 1 Algoritmo di Viterbi

Require: observation on len T
Require: state-graph on len N
  create a path probability matric viterbi[N+2, T]
  for each state s from 1 to N **do**
    $viterbi[s, 1] \leftarrow a_{0,s} * b_s(o_1)$
    $backpointers[s, 1] \leftarrow 0$
  for each time step t from 2 to T **do**
    for each state s from 1 to N **do**
      $viterbi[s, t] \leftarrow max_{s'=1}^{N} viterbi[s', t-1] * a_{s',s} * b_s(o_t)$
      $backpointer[s, t] \leftarrow argmax_{s'=1}^{N} viterbi[s', t-1] * a_{s',s}$
  $viterbi[q_F, T] \leftarrow max_{s=1}^{N} viterbi[s, T] * a_{s,q_F}$
  $backpointer[q_F, T] \leftarrow argmax_{s=1}^{N} viterbi[s, T] * a_{s,q_F}$
  return best-path

L'algoritmo di Viterbi sfrutta la programmazione dinamica per conservare soltanto il path con maggiore probabilità

Le HMM possono essere estese a n-grammi allungando la storia, ma si rischia la sparsity

La sparseness può essere combattuta combinando le probabilità degli n-grammi

$$P(t_i|t_{i-1}t_{i-2}) = \lambda_3\hat{P}(t_i|t_{i-1}t_{i-2}) + \lambda_2\hat{P}(t_i|t_{i-1}) + \lambda_1\hat{P}(t_i)$$

Dove $\lambda_1 + \lambda_2 + \lambda_3 = 1$

Le HMM non sono molte informative sulle feature

4.2.5 MEMM (maximum entropy markov models)

Modelling Identifichiamo set di feature eterogenee e le combinaiamo in modo probabilistico

L'obiettivo è avere una distribuzione più uniforme possibile

$$P(y|x)? = \sum_{i=1}^{N} w_i f_i$$

$$? = w \cdot f$$

$$p(y = c|x) = p(c|x) = \frac{1}{Z} exp(\sum_i w_i f_i)$$

In questo modo si possono combinare feature a piacere

Learning Il learning viene fatto tramite la multinomial logistic regression

$$\hat{w} = argmax_w \sum_j logP(y^{(j)}|X^j)$$

Si riduce a un problema di ottimizzazione convessa (hill-climbing e SGD)

Decoding

$$v_t(j) = max_{i=1}^{N} v_{t-1}(i) P(s_j|s_i, o_t), 1 \leq j \leq N, 1 < t \leq T$$

4.2.6 NER tagging

Named-Entities tagging, 4 tag principali, PER, LOC, ORG, GPE, si usano per parole multiple che sono una singolo frammento di significato

Questo task è più difficile del PoS tagging, bisogna segmentare il testo e disambiguare

BIO tagging

Beginning inside outside algorithm, abbiamo 1 tag O, n tag B e n tag I

Capitolo 5

Sintassi

Capacità di comprendere la struttura di una frase, ovvero la relazione tra le parole

Dal livello morfologico il parsing sintattico ricostruisce l'albero della frase (può essere sia a costituenti che a dipendenze)

La struttura dati di riferimento è l'**albero**

5.1 Grammatiche generative

Si distinguono due aspetti fondamentali, la competence e la performance

COMPETENCE: grammatica formale

PERFORMANCE: algoritmo di parsing

Le grammatiche di Chomsky sono formate da $G = (\Sigma, V, S, P)$, ovvero alfabeto, insieme dei simboli non terminali, token di start e regole di riscrittura/produzione

Costituenti == simboli non terminali della grammatica

Chomsky teorizza una gerarchia di grammatiche (al crescere dell'espressibità cresce la complessità)

Non tutte le lingue sono CF, ad esempio il tedesco svizzero ha frasi con dipendenze incrociate/cross serial

Per lingue simili bisogna usare grammatiche mildly context sensitive, tree adjoining grammars, combinatory categorial grammars

Queste grammatiche sono tutte equivalenti tra loro

5.2 Parsing a costituenti

Rappresenta il raggruppamento di parole in una stessa relazione

La struttura della frase è rappresentata in forma annidata

Un costituente **agisce come una unità nella frase** (HP di Loebell), è un gruppo di parole contigue nella frase

Chomsky ipotizza di poter rappresentare la grammatica inglese come Context Free Grammar a patto che i simboli non terminali siano i costituenti della lingua

Teoria X-barra, permette di passare dalla struttura a dipendenze a quella a costituenti e viceversa

Viene spesso chiamato **albero a sviluppo verticale**

5.2.1 Mildly context sensitive

Create da Joshi nel 1985, superset delle CF, permettono dipendenze nested e cross-serial

Parsiicabili in tempo polinomiale

Hanno la proprietà di crescita costante: se estendo una frase corretta ottengo ancora un frase valida

Complessità $a^n b^n c^n$

5.2.2 Tree adjoining grammars

Joshi et al. 1975

Idea fondante: innesto di sottoalberi

Lexicalized tree adjoining grammars

Lessicalizzazione della grammatica

Fattorizza la ricorsione con operazioni di adjoining

Estende il dominio della Località

5.2.3 Head grammars

Pollard 1984

5.2.4 Linear indexed grammars

Gazdar 1985

5.2.5 Combinatory categorial grammars

Steedman 1985 e Satta 2010

Approccio bottom up

5.2.6 Anatomia dei parser

I parser sono caratterizzati da:

- Una grammatica (CF, TAG, CCG, Dependency, ...)
- Algoritmo, con la sua strategia di ricerca e l'organizzazione della memoria
- L'oracolo (Probabilistico, basato a regola, ...)

La strategia di ricerca è caratterizzata dal verso in cui si legge la stringa in input e dalla strategia di costruzione della frase, Top-Down o Bottom-Up

La strategia top-down è goal-directed, partendo dalla radice S si generano tutte le frasi possibili, ma si generano anche frasi non compatibili con le parole. Questo approccio è definito **Razionalista**

La strategia bottom-up è data-directed, partendo dalle parole si cerca di risalire i possibili alberi, ma si generano anche alberi non corretti (radice diversa da S). Questo approccio è definito **Empiristico**

5.2.7 Parser A

Usa grammatiche CF

Algoritmo Top-Down, left-to-right e con backtracking

Oracolo a regole

I parser non prevengono l'ambiguità, dipende dalla grammatica, una frase può avere più alberi validi

Inoltre ci sono ambiguità legate alla coordinazione, all'attachment e strutturali

L'ambiguità strutturale è espressa dai numeri catalani, una serie potenza $C_n = \prod_{k=2}^{n} \frac{n+k}{k}$

L'esplosione combinatoria dei possibili alberi si può gestire con la programmazione dinamica

5.2.8 Parser B: CKY - Cocke Kasami Younger

Calcola tutti i possibili parse in $O(n^3)$, richiede una grammatica CNF e il caso medio coincide con il caso peggiore

Usa grammatiche CF con parser bottom-up, left-to-right e la programmazione dimanica per gestire la memoria

L'oracolo è a regole

Algorithm 2 CKY-parse

Require: words
Require: G
  for j $From$ 1 To $LENGTH(words)$ **do**
    for $A|A \rightarrow words[j] \in G$ **do**
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for i $From$ $j-2$ To 0 **do**
      for k $From$ $i+1$ To $j-1$ **do**
        for $A|A \rightarrow BC \in G \wedge B \in table[i,k] \wedge C \in table[k,j]$ **do**
          $table[i, j] \leftarrow table[i, j] \cup A$
  return table

Algoritmo L'indice j serve per effettuare il loop sulle colonne della matrice e riempire le celle in basso

L'indice i permette di riempire l'i-esima riga della colonna j

L'indice k serve per fare loop tra le possibili produzioni

L'algoritmo è troppo lento per usi real-time, per cui si opta per pruning probabilistico, risultati parziali o si adatta per lavorare sulle dipendenze

5.2.9 Parser C: CKY probabilistico

Usa grammatiche CF probabilistiche

Ricerca bottom-up, left-to-right

Programmazione dimanica e beam search

Oracolo probabilistico

La probabilità dell'abero è data dal prodotto delle probabilità delle produzioni

Le probabilità sono calcolate dai TreeBank

I TreeBank definiscono in modo induttivo anche le grammatiche (contengono solo produzioni valide)

$$P(\alpha \rightarrow \beta|\alpha) = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

Algoritmo

5.2.10 Parser D: CKY-neurale

La grammatica è prodotta da una rete neurale

Parsing bottom-up, left-to-right

Algorithm 3 PROBABILISTIC-CKY

Require: words
Require: G
  for j $From$ 1 To $LENGTH(words)$ **do**
    for $A|A \rightarrow words[j] \in G$ **do**
      $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
    for i $From$ $j-2$ To 0 **do**
      for k $From$ $i+1$ To $j-1$ **do**
        for $A|A \rightarrow BC \in G \wedge table[i,k,B] > 0 \wedge table[k,j,C] > 0$ **do**
          if $table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
  then
            $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
            $back[i,j,A] \leftarrow k, B, C$
  return BUILD-TREE(back[1, WORDS, S]), table[1, LENGTH(words), S]

Memoria organizzata tramite programmazione dimanica e beam search
Oracolo probabilistico
Valutato tramite precision, recall, F-score

$$Precisione = \frac{T^+}{T^+ + F^+}$$

$$Recall = \frac{T^+}{T^+ + F^-}$$

La valutazione è fatta sulla base dei subtree (rende la metrica più robusta)
Si possono anche usare grammatiche CF lessicalizzate, in cui si associa anche la testa del constituente, in modo da rendere le probabilità più precise (forte shift verso la semantica)

5.2.11 Parser E: chunk parsing a regole

Grammatiche regolari a cascata
Algoritmo bottom-up con programmazione dimanica
Oracolo rule based
Strutturo la frase in unità sintattiche → chunk
In questo modo si elimina la ricorsione
Le grammatiche regolari messe a cascata sono in grado di approssimare le CFG
La versione probabilistica usa l'algoritmo IOB per trovare i chunk

5.3 Parsing a dipendenze

Si concentra sulle relazioni tra coppie di parole (relazione di dipendenza asimmetrica)

Le relazioni sono formate da due parti, la **Testa**, ovvero la parola dominante e la **Dipendente**, ovvero la parola dominata

La testa della relazione sceglie i suoi dipendenti e determinano le proprietà

Viene spesso chiamato **albero a sviluppo orizzontale**

Ci sono relazioni semplici (come tra il verbo e il suo soggetto/oggetto) e relazioni difficili/problematiche (come con la punteggiatura)

Nell'ipotesi di Tesnière si fanno 2 approssimazioni, *di* è una funzione che trasforma un nome in aggettivo, *e* rende la relazione simmetrica

5.3.1 Grammatiche a dipendenza, 2019

Marneffe e Nivre introducono le grammatiche a dipendenze

Vantaggi

- Sono più generiche
- Più simili al funzionamento umano
- Trasparenti e semplici nella rappresentazione

Rendono l'uso libero dall'ordine delle parole nella frase

Approcci per il parsing a dipendenze:

- Programmazione dimanica (simile a CKY)
- Algoritmo delle PCFG lessicalizzate $O(n^5)$
- Nel 1996 Eisner ha scoperto un algoritmo per ridurre la complessità a $O(N^3)$

Tecniche per il dependency parsing:

- Grafi in cui si trova MST
- Parsing a costituenti + conversione
- Parsing deterministico, scelta greedy guidate dal Machine learning
- Soddisfacimento di vincoli

5.3.2 Parser F: MALT

Parsing deterministico a trasizioni

Usa grammatiche a dipendenze (ricavate da TreeBank)

Algoritmo bottom-up con ricerca depth first

Oracolo probabilistico

Malt prevede:

- uno stack di lavoro
- un input buffer
- un parser con oracolo che associa l'azione (leftarc, rightarc, shift)
- queste componenti creano le dipendenze tra le parole della frase

Con questa struttura si effettua *Transition based parsing*

Uno stato è composto da 3 elementi, stack, lista di parole rimanenti e insieme delle dipendenze create fino a quel punto

Operazione shift Prende la prossima parola e la mette sullo stack rimuovendola dalla lista

Operazione left Crea una dipendenza (a, b) tra la prossima parola della lista (a) e la parola in cima allo stack (b)

Poi fa pop dallo stack

Operazione right Crea una dipendenza (b, a) tra la prossima parola della lista (a) e la parola in cima allo stack (b)

Rimuove a dalla lista

Fa pop dallo stack e mette la parole a inizio lista

Definizione di proiettività Con un parser del genere è possibile controllare in modo semplice la proiettività

G è proiettivo

Se $i \rightarrow j$ allora $i \rightarrow *k$ per ogni k tale che $i < k < j$ o $j < k < i$

La non proiettività è necessaria per gestire dipendenze a lungo raggio (come per le domande) e le dipendenze cross seriale (nelle mildly context sensitive language)

5.3.3 Problemi dell'oracolo

- Come scelgo l'etichetta delle relazioni
- Come scelgo l'operazione

Il primo problema si risolve usando al posto che 3 operazioni 2n+1, dove n è il numero di etichette (n per left, n per right e 1 per shift)

Per il secondo posso creare un oracolo a regole (scelgo l'operatore in base alle regole)

MALT usa machine learning per addestrare un classificatore e scegliere l'operatore

feature per addestrare l'algoritmo di machine learning Stato + attributi delle parole (scelti in base alla conoscenza di dominio)

Oppure appresi tramite rete neurale

training data Si possono usare TreeBank (già a dipendenze o convertiti)

training algorithm Dato il TreeBank costruisco le scelte del parser (con le operazioni corrette)

- Left se trovo una dipendenza corretta nell'albero
- right se trovo una dipendenza corretta nell'albero E tutte le altre dipendenze associate alla parola figlia si trovano già nell'albero
- Shift altrimenti

5.3.4 Parser G: TUP

Grammatica a dipendenze con vincoli

Algoritmo bottom-up e depth first

Oracolo rule-based

Usava regole per chunking (nominale e non verbale), coordination (rimuove ambiguità delle congiunzioni con delle regole), verb-subcat (regole verbali basate su tassonomia di classi per la sottocategorizzazione)

Dipendenze morfo-sintattiche, sintattico-funzionali e semantiche

Capitolo 6

Semantica

Comprensione del significato delle singole parole e del significato complessivo della frase

Bisogna capire che struttura ricostruiamo nella nostra testa per rappresentare la conoscenza e poterci ragionare sopra

Le strutture dati di riferimento sono gli insiemi (synset) e i vettori (embeddings)

Inizialmente si costruivano grammatiche a mano (Lesmo per l'italiano, SHRDLU di winograd o CHAT-80 per l'inglese)

I primi approcci usavano la programmazione logica (come ProLog) per descrivere la realtà

Il problema è che FoL non può gestire frasi incomplete, si passa quindi a Lambda-FoL

Un approccio fondazionale per la semantica computazionale è:

- parsificare la frase e ottenere l'albero a costituenti
- cercare la semantica di ogni parola
- costruire la semantica per ogni sintagma in modo bottom-up e syntax driven (a regole)

Questo procedimento funziona per il principio di composizionalità di Frege: *Il significato di una frase è dato dal significato delle parti e di come queste sono combinate*

Per Montague i linguaggi naturale non sono diversi da quelli formali (a patto di usare un formalismo sufficientemente ricco)

6.1 λ abstraction

Viene usato l'operatore λ per fare binding di variabili libere
(es. $\lambda x.love(x, francesca) \rightarrow amare\ francesca$)

Per usare λ-FoL si passa al formalismo delle **augmented CFG**, ovvero grammatiche ad attributi

Tramite beta riduzione si applica la funzione che sostituisce la variabile libera con l'oggetto istanziato

Beta-reduction

- Elimino λ
- rimuovo l'argomento
- sostituisco la variabile libera con l'argomento

6.1.1 Problemi semantica computazionale

I quantificatori possono generare ambiguità, ad esempio il quantificatore esistenziale può essere visto come un articolo indeterminativo oppure "c'è" + articolo indeterminativo

Questa situazione si gestisce facendo reverse engineering e permettendo l'astrazione anche sui predicati

Gestendo l'astrazione sugli articoli è necessario ripensare alla gestione dei nomi propri, si usa *type-raising*, in modo analogo si gestiscono anche i verbi transitivi

I punti di forza delle astrazioni λ risiedono nel fatto che, a patto di considerare in modo abbastanza generico la grammatica, aggiungono sistematicità nel trattamento delle parti della frase

Inoltre tramite reificazione è possibili generalizzare le funzioni della nostra grammatica FoL e renderle molto generiche e robuste all'aggiunta di avverbi

6.2 Semantica lessicale (fase 1)

Connessioni legate al significato dei vari lessemi (coppia forma-signiicato == elemento del lessico)

Problema, nel vocabolario la definizione delle parole è spesso dipendente dalla parola stessa

Per questo motivo le risorse semantiche spesso si concentrano su relazioni come omonimia (due parole con stessa forma ma significato diverso),

polisemia (stessa parola con più significati), sinonimia (due parole morfologicamente diverse hanno stessa forma), iponimia (relazione di sottoclasse)...

Ci sono due approcci principali per la semantica lessicale, classico e distribuzionale

6.2.1 Classico

6.2.2 Distribuzionale

Le parole sono rappresentate da vettori (embeddings)

Problema, i vettori lunghi possono essere sparsi (rappresentazione one-hot). Si può ovviare con vettori più corti e quindi densi, più informativi per i sistemi di machine learning

L'approccio distribuzionale a sua volta si divide in modelli statistici e modelli neurali

Statistico

Neurale

Imparano i vettori dai dati, le feature sono implicite

Ad oggi sono molto usati gli embeddings contestuali, ovvero che contengono parte dell'informazione del contesto, aiutano ad associare il senso corretto al termine

6.3 Semantica composizionale (fase 2)

Rappresentazione della conoscenza per poter fare ragionamento

Ispirata alla semantica dei linguaggi formali

HP fondamentale: la semantica di un sintagma è funzione della semnatica dei sintagmi componenti e non dipende da sintagmi esterni

Un numero finito di regole controlla un numero infinito di situazioni

Nel 1974 Montague propone il linguaggio come forma logica (FOL + lambda calcolo) $\rightarrow$ Questo permette ragionamento

La semantica composizionale + la semantica lessicale permettono di dare un senso anche alle parole sconosciute (es. metasemantica con la gnosi delle fanfole di fosco maraini)

Capitolo 7

Pragmatica

Conoscenza sul mondo che permette di usare il linguaggio in modo contestuale

I rapporti pragmatici sono relazioni tra elementi di una frase e elementi che potrebbero essere intercambiati nella frase stessa

L'interpretazione di alcune parti di una frase può dipendere da informazioni circostanziali date per scontato quando si comunica

Anafora: sintagmi che si riferiscono a oggetti menzionati in precedenza

La struttura dati di riferimento sono i **frame** (bisogna valutare se aggiungere o meno la ricorsione). Inoltre si usano ontologie per il ragionamento

Capitolo 8

Discorso

Capacità di conversare facendo domande pertinenti

La struttura dati di riferimento sono i **frame** (bisogna valutare se aggiungere o meno la ricorsione). Inoltre si usano ontologie per il ragionamento

8.1 Natural language generation

Definizione di McDonald, 1992: Processo di costruzione di un testo in linguaggio naturale per venire incontro a specifici obiettivi comunicativi

Il sistema ha in input una rappresentazione generica dell'informazione, che può provenire da qualsiasi origine

In output deve restituire documenti, spiegazioni, messaggi

Le basi di conoscenza in cui si muove possono essere basate sul linguaggio o sul dominio

NLG è stato usato con molto successo nel caso di **BabyTalk**. I bambini nei reparti sono sorvegliati con una pletora di sensori, ma i dati prodotti sono difficili da leggere, per questo motivo si usava NLG per produrre un riassunto. Inoltre il sistema produceva 3 riassunti per 3 target diversi, uno ogni 45min per i dottori, uno ogni 12 ore per le infermiere, uno ogni 24 ore per i genitori

La valutazione di BabyTalk è stata fatta sia basandosi su task che basandosi su valutazione umana

NLG vs. template I template sono "text planning" e permettono una formalizzazione a basso livello, non è necessaria linguistica avanzata e sono veloci

Tuttavia NLG permette di migliorare la qualità del testo prodotto, di generare l'output in più lingue ed è più mantenibile

Ad oggi si usano sistemi ibridi (o NLG nei template o template nell'NLG, come in MADiMAN)

BigData NLG funziona bene in tutti quei contesti in cui ci interfacciamo con BigData, per via della loro mole è impossibile leggerli singolarmente, NLG permette un'analisi senza perdita di informazione ma gestibile da operatori umani

Problemi Si possono usare strutture a template, ma bisogna catturare strutture molto complesse di sintassi e la ricorsione

Uno "shortcut" (proposto nel 2000 da Mellish) è rendere i moduli triviali evitandoli o scegliendo gli operatori da usare in base a cosa si aspettano i moduli

8.1.1 Task con NLG

Content determination Si decide cosa dire

Discourse planning Si mettono in relazione le frasi (tramite conceptual grouping e rhetorical relationships)

Sentence aggregation Le singole frasi si devono combinare tra loro per creare un testo coeso e fluido

Il risultato è una sentence specification o un sentence plan

Lexicalisation A questo punto si decidono le parole da usare per esprimere concetti e relazioni del dominio e le relazioni sintattiche tra le parole

Si decidono le sfumature di significato da dare ad una parola

C'è il problema del *Lexical Gap* per lingue diverse

Referring expression generation Si decidono quali parole usare per esprimere entità, bisogna bilanciare fluenza e ambiguità

Syntactic and morphological realization Si applicano le regole morfologiche e di sintassi per avere frasi corrette

Orthographic realization Si aggiunte la capitalizzazione delle parole, la punteggiatura, la formattazione

8.1.2 Architettura standard

3 blocchi principali

Text Planning in cui ho Content determination e Discourse planning
In questa fase si fa ampio uso della conoscenza di dominio

Sentence Planning in cui effettuo Sentence aggregation, Lexicalisation e Referring expression generation
Viene usata sia la conoscenza di dominio che quella sulla lingua di arrivo

Linguistic Realization Syntactic and morphological realization + Orthographic realization
Ampio uso di conoscenze linguistiche

8.1.3 Sistemi di dialogo e ChatBot

Dialogo: attività collaborativa
Attività

- Motivata da goal
- Soggetto a costi di minimizzazione dello sforzo comunicativo
- Composto da una sequenza temporale di eventi (turni)

Collaborativo

- Gli agenti devono coordinare i contributi alla conversazione
- Bisogna assicurare compresione mutua
- Bisogna rendere l'interazione efficiente (ECONOMIA LINGUISTICA)

Ci sono 4 key-feature nel dialogo

Turni Turni conversazionali
Si riduce al problema di allocazione di risorse tramite il canale conversazionale
Empiricamente si è misurato che il tempo per il cambio di turno è di circa 250ms
Indizi per il cambio:

- Silenzio

- Esitazione
- Sintassi (parti completate)
- Intonazione
- Intensità
- Linguaggio del corpo

Solitamente i sistemi hanno iniziativa mista e un metodo comune per il cambio di turni è aspettare 0.5/1s

Si costruisce una comunicazione Ping-Pong, senza interruzioni e sovrapposizioni

Dialogue acts I turni sono costituiti da unità comunicative di base (dialogue acts/speech acts)

Per Searle (1975):

- Assertivi: si parla di verità
- Direttivi: ordini/richieste
- Commissivi: si prende in carico un compito
- Espressivi: esprimono sentimenti/stati d'animo
- Dichiarativi: dichiarano un cambio nello stato delle cose

Contesto L'interpretazione degli atti di dialogo è dipendente dal contesto

3 punti chiave:

- Non-sentential utterances: parti del dialogo che non hanno significato prese singolarmente (es. ellissi)
- Implicature conversazionali: parti di dialogo che non sono conseguenze dirette della parte precedente ma collegate dal contesto
- Espressioni di riferimento (es. modi di dire o parti di comunicazione non espresse tramite il parlato, uso della pragmatica)

Grice nel 1989 espone 4 massime per le implicature conversazionali:

- Quantità
- Qualità
- Relazione (essere significativi)
- Significato (essere chiari)

Segnali di grounding I partecipanti si scambiano segnali di grounding che determinano la comprensione (o la mancanza di comprensione) di alcune parti

8.1.4 Architetture per CB e DS

Chatbot ELIZA Rule-based nella sua versione originale

ELIZA interpretava uno psichiatra/psicologo

Usava diverse regole per analizzare la frase e generare la risposta tramite template

Le regole erano gestite per classi e si impediva di usare la stessa regola in modo ripetuto (rendeva l'esperienza più verosimile ed evitava loop)

EFFETTO PAREIDOLIA, l'assistente di Weizenbaum chiedeva del tempo in privato con la macchina (vedeva l'uomo dove non c'era)

PARRY Espansione di ELIZA, gestiva variabili, controllava le strutture e aveva capacità di comprensione, primo sistema a passare il test di Turing

ALICE ELIZA + PARRY

Artificial Intelligence Markup Language

Permetteva di definire regole per le risposte

Si usano tag tipo HTML

CB su corpus In alternativa i CB possono essere ottenuti tramite corpus (information retrieval o NN encoder-decoder)

Il problema dei chatbot di questo tipo è che possono solo emulare il training set

8.1.5 Architettura dei sistemi di dialogo

L'umano parla, si passa al sistema di **speech recognition**, il testo compreso viene passato al modulo di **language understanding** e il risultato è l'input per il **dialogue manager**. Il **dialogue manager** è composto da due moduli, il **dialogue control** e il **dialogue context model** e tramite essi genera un output per **response generation**, il testo prodotto viene passato tramite **text to speech synthesis**

Speech recognition Le perfomance variano in base a:

- dimensioni del vocabolario

- speaker con cui si interfaccia
- fenomeni di coarticolazione
- se lo speaker sta leggendo un testo o conversando
- la bontà del sistema
- il microfono utilizzato

Language understanding Task complesso, c'è molta ambiguità

Nel metodo classico si usava il sistema a cascata sintassi, semantica e pragmatica

Adesso si tende a usare una semantica semplice e l'approccio a frame

Nel 1977 GUS è stata la prima proposta di sistema di dialogo a frame/solt

Questi sistemi sono ancora molto usati per i bot aziendali

I sistemi a frame sono molto precisi e funzionano bene in domini ristretti, di contro sono costosi e richiede tempo creare le regole, inoltre hanno forti problemi di recall

Dialogue manager Ci sono due paradigmi principali (Larsson 2005), il primo vede il DS come front-end, il secondo come agente

Dialogue context model Mantiene le informazioni importanti e le estende con quanto detto fino a quel momento e con quello di cui si è già fatto grounding

Si usano grafi, frame, piani/logica o modelli statistici

I sistemi a grafi sono utili per interazioni semplici (in caso contrario la dimensione esplode e diventa intrattabile)

I sistemi a frame sono utili per domini limitati, ma possono gestire molte variazioni

Dialogue control Decide cosa fare successivamente (chiedere maggiori informazioni, chiarificare, fare grounding)

Si usano stati finiti, slot tipati e formule FoL

Response generation Processo in 3 fasi con Text planning, Sentence planning e Linguistic realization

Text to speech synthesis Si può generare il parlato oppure usare lo standard HTML del W3C e il tag *phoneme*